1 Projeto – Lógica Matemática

Desenvolva um programa que realize as operações de uma **fórmula lógica**, no formato de **tabela-verdade**.

1.1 Python Truth Table

A linguagem recomendada é o **Python**, que oferece a biblioteca **truth-table-generator**, preparada para realizar as operações de uma **fórmula lógica**, no formato de **tabela-verdade**.

Referências Python: **truth-table-generator**

- Site: https://pypi.org/project/truth-table-generator/
- Instalação: **pip install truth-table-generator**
- Operações admitidas e suas representações:

1. Negação: `'not'`, `'-'`, `'~'`
2. Disjunção / OU lógico: `'or'`
3. NOR: `'nor'`
4. XOR / OU exclusivo: `'xor'`, `'!='`
5. Conjunção / E lógico: `'and'`
6. NAND: `'nand'`
7. Implicação / Condicional: `'=>'`, `'implies'`
8. Bicondicional: `'='`

Exemplos de utilização

1) Exemplo básico

```
print(ttg.Truths(['p', 'q', 'r'], ['p and q and r', 'p or q or r','(p or (~q)) => r']))
+-----+-----+-----+-----------------+---------------+--------------------+
|  p  |  q  |  r  |  p and q and r  |  p or q or r  |  (p or (~q)) => r  |
|-----+-----+-----+-----------------+---------------+--------------------|
|  1  |  1  |  1  |        1        |       1       |         1          |
|  1  |  1  |  0  |        0        |       1       |         0          |
|  1  |  0  |  1  |        0        |       1       |         1          |
|  1  |  0  |  0  |        0        |       1       |         0          |
|  0  |  1  |  1  |        0        |       1       |         1          |
|  0  |  1  |  0  |        0        |       1       |         1          |
|  0  |  0  |  1  |        0        |       1       |         1          |
|  0  |  0  |  0  |        0        |       0       |         0          |
+-----+-----+-----+-----------------+---------------+--------------------+
```

2) Exemplo com True / False

```
print(ttg.Truths(['p', 'q'], ['p and q', 'p or q', '(p or (~q)) => (~p)'], ints=False))
+-------+-------+-----------+----------+-----------------------+
|   p   |   q   |  p and q  |  p or q  |  (p or (~q)) => (~p)  |
|-------+-------+-----------+----------+-----------------------|
| True  | True  |   True    |   True   |         False         |
| True  | False |   False   |   True   |         False         |
| False | True  |   False   |   True   |         True          |
| False | False |   False   |  False   |         True          |
+-------+-------+-----------+----------+-----------------------+
```

3) Usando variável

```
table = ttg.Truths(['p', 'q'], ['p => q', 'p = q'])
print(table)
```

```
+-----+-----+----------+---------+
|  p  |  q  |  p => q  |  p = q  |
|-----+-----+----------+---------|
|  1  |  1  |    1     |    1    |
|  1  |  0  |    0     |    0    |
|  0  |  1  |    1     |    0    |
|  0  |  0  |    1     |    1    |
+-----+-----+----------+---------+
```

2 Requisitos do Projeto

1. Fazer a **leitura de uma fórmula** como string;
2. Garantir que as **variáveis proposicionais** da fórmula serão apresentadas em **letras minúsculas**;
3. Apresentar uma **lista, sem repetição, das variáveis proposicionais passadas na fórmula**, em **ordem alfabética**;
4. Identificar quais **as operações são usadas na fórmula** e **a quantidade de vezes** que elas ocorrem;
5. Apresentar a **tabela verdade correta** da fórmula;
6. **Perguntar se o usuário quer encerrar o programa ou continuar** com nova fórmula. Ao escolher continuar, o programa volta a solicitar nova fórmula e apresentar todos os resultados, em um laço de repetição, até que o usuário encerre o programa.

3 Entrega

3.1 Artefatos

1. **Vídeo** (MP4 ou links no OneDrive, GoogleDocs ou Youtube Não Listado) para apresentar e defender o trabalho: de 5 a 10 minutos de gravação
    - Cada **integrante da equipe deve apresentar uma parte do trabalho** no vídeo
        i. Deve apresentar seu nome na legenda do vídeo
        ii. Deve aparecer no vídeo E / OU se apresentar
        iii. Deve apresentar parte do trabalho (pode ser em off, apenas narrando)
2. **Código Fonte em arquivo**
    - Documentado / comentado (explicação do funcionamento de uma função / método do trabalho, de acordo com o comentário).

3.2 Roteiro do Vídeo

Cada integrante da equipe deve:

1. Se apresentar (**no vídeo, o nome do estudante também aparece como legenda**).
2. Mostrar a inserção de uma **fórmula**.
3. Mostrar que os **requisitos solicitados** estão aparecendo no resultado gerado.
4. **Apresentar** o código e **explicar** como o requisito foi atendido:
    I. Mostrar a leitura de uma fórmula como string;
    II. Mostrar a como as variáveis proposicionais da fórmula são apresentadas em letras minúsculas e ordem alfabética;
    III. Mostrar como a lista das variáveis proposicionais da fórmula foi criada sem repetição;
    IV. Mostrar como são identificadas as operações usadas na fórmula, juntamente com a quantidade de vezes que as operações ocorrem;
    V. Mostrar como a tabela verdade foi gerada, e também que a tabela está correta;
    VI. Mostrar como o usuário decide se quer encerrar o programa ou continuar com nova fórmula.

3.3 Exemplos resultados obtidos

```
 -------------------------------------------------------
 -- Digite a formula: not(Q or p) => ~R

 -- Formula em minúsculo:  not(q or p) => ~r

 -- Variáveis:  ['p', 'q', 'r']

 -- Operações:
       not:   2 vezes
        or:   1 vez
    implic:   1 vez
+-------+-------+-------+---------------------+
|   p   |   q   |   r   |  not(q or p) => ~r  |
|-------+-------+-------+---------------------|
| True  | True  | True  |        True         |
| True  | True  | False |        True         |
| True  | False | True  |        True         |
| True  | False | False |        True         |
| False | True  | True  |        True         |
| False | True  | False |        True         |
| False | False | True  |        False        |
| False | False | False |        True         |
+-------+-------+-------+---------------------+


Deseja continuar? (s/n)s

 -------------------------------------------------------
 -- Digite a formula: ~((T => R) and (r => t))

 -- Formula em minúsculo:  ~((t => r) and (r => t))

 -- Variáveis:  ['r', 't']

 -- Operações:
       not:   1 vez
       and:   1 vez
    implic:   2 vezes
+-------+-------+-----------------------------+
|   r   |   t   |  ~((t => r) and (r => t))   |
|-------+-------+-----------------------------|
| True  | True  |           False             |
| True  | False |            True             |
| False | True  |            True             |
| False | False |           False             |
+-------+-------+-----------------------------+


Deseja continuar? (s/n)s
```

```
 ---------------------------------------------------------
 -- Digite a formula: p nand P

 -- Formula em minúsculo:  p nand p

 -- Variáveis:  ['p']

 -- Operações:
      nand:   1 vez
+-------+------------+
|   p   |  p nand p  |
|-------+------------|
| True  |   False    |
| False |    True    |
+-------+------------+


Deseja continuar? (s/n)s

 ---------------------------------------------------------
 -- Digite a formula: (~p or q) and (p => q)

 -- Formula em minúsculo:  (~p or q) and (p => q)

 -- Variáveis:  ['p', 'q']

 -- Operações:
       not:   1 vez
        or:   1 vez
       and:   1 vez
    implic:   1 vez
+-------+-------+--------------------------+
|   p   |   q   |  (~p or q) and (p => q)  |
|-------+-------+--------------------------|
| True  | True  |           True           |
| True  | False |          False           |
| False | True  |           True           |
| False | False |           True           |
+-------+-------+--------------------------+


Deseja continuar? (s/n)s

 ---------------------------------------------------------
 -- Digite a formula: (P => Q) and (q => p)

 -- Formula em minúsculo:  (p => q) and (q => p)

 -- Variáveis:  ['p', 'q']

 -- Operações:
       and:   1 vez
    implic:   2 vezes
+-------+-------+-------------------------+
|   p   |   q   |  (p => q) and (q => p)  |
|-------+-------+-------------------------|
| True  | True  |          True           |
| True  | False |          False          |
| False | True  |          False          |
| False | False |          True           |
+-------+-------+-------------------------+
```

4 Ajuda para confecção do Programa em Python

```
vars = sorted(vars) # vars = lista com as variáveis em ordem alfabética
```

```
opers = {}
opers.update({'not': 3}) # opers = dictionary key-value

# { 'not': 3 , 'or': 2 }
```

```
string = 'aba aba aba'
x = string.count('aba')
print(f"The word '{string}' contains the substring 'aba' {x} times.") }
```